Reflecting on classroom practice: Spatial reasoning and simple coding

Alessandra King
Holton-Arms School, USA
<alessandra.king@holton-arms.edu>

Introduction

Spatial reasoning and computer programming are precious skills in our highly technological world. An activity that combines the two in an engaging way may be very valuable. As an introduction to more formal geometry units, in my pre-algebra class we like to play games with blocks and cards to hone spatial skills because, "Students' skills in visualising and reasoning about spatial relationships are fundamental in geometry" (NCTM, 2000, p.237). Geometric reasoning is also identified as an important topic in the Measurement and Geometry content strand of the *Australian Curriculum: Mathematics* (ACARA, 2014), and appears one way or the other in all four proficiency standards of Understanding, Fluency, Problem Solving and Reasoning.

Spatial reasoning—the ability to visualise and play with shapes in one's mind—is essential in many fields, and crucial in any Science, Technology, Engineering, Mathematics [STEM] discipline. It is, for example, the ability that the engineer needs to build bridges; the chemist to see the three-dimensional structure of a molecule; the architect to design buildings; and, the doctor to navigate the patient's abdominal cavity with a laparoscope. Spatial thinking is an important factor for achievement in areas of STEM. And while there is evidence of a gender gap in this area, there is also evidence that these reasoning skills can be improved with practice.

Instead of using manipulatives, I guided my students to work on these skills with a short unit on coding. Such a unit is not intended to teach computer programming, or substitute for curricular geometry units: it aims only to expose children to some coding (in order to raise their interest in it for possible future studies), and to supplement the regular geometry curriculum by engaging the students in some spatial reasoning exercises.

The demand to add or focus on STEM subjects in primary schools has grown world-wide and in many countries, for example the United Kingdom, Germany, and Sweden. There is a strong drive in these nations to introduce programming to all students. In Australia, consideration has been given to a new digital technology curriculum that will expose students to coding in Year 5 and require them to start programming in Year 7. The connection between mathematics and technology is clearly stated in the *Digital Technology* curriculum (ACARA, 2014a) which is currently awaiting final endorsement. It states that, "The Technologies curriculum provides contexts within which mathematics understanding, fluency, logical reasoning, analytical thought and problem-solving skills can be applied and developed", and "Students use spatial understandings developed in mathematics to apply knowledge of geometry, shapes and angles in Technologies". In the United States news, reports indicate that the country will continue to suffer a serious shortage of software professionals for the foreseeable future, and in many other fields familiarity with computer programming and

data manipulation will be increasingly in demand. In addition, females have long been under-represented in computer science and not enough girls are choosing to study it. For example, while girls are the majority of the students taking Advanced Placement Program ® (AP) examinations, they were outnumbered 4 to 1 in the AP Computer Science examination in 2014, according to the data released by the College Board (*The College Board*, 2014). Although girls are interested in the STEM fields at a younger age, there is evidence that their interest declines more substantially (University of Idaho, 2015); so they need to be encouraged to see that coding can actually be a lot of fun.

Since I teach in an all-girls school, I am addressing both the spatial reasoning and coding gap with a brief unit I created. This unit was designed to simultaneously introduce my students to a beginner level of computer programming, and assist them to develop and practice their spatial skills.

Learning objectives

The primary learning objectives for this activity were to assist my students to practice spatial thinking in a new and exciting way. In addition, I was looking to provide a gentle and fun introduction to coding. Both objectives are highly valuable in themselves. "Facility with geometric thinking is essential to success in the later study of mathematics and also in many situations that arise outside of the mathematics classroom" (NCTM, 2000, p.210). Moreover, giving students even a simple introduction to programming supports them in understanding what happens inside a computer and how computers communicate, making technology more approachable and less obscure. Finally, programming teaches other important skills like logical thinking, problem solving, and perseverance, as well as collaboration and communication.

Project design

Before starting the class activity, I looked at various tools. I had a few criteria: first, it had to be a tool that worked on the iPad, as this is the device we use in our Middle School. Secondly, it had to allow students to move at their own pace and to appreciate their own progress; and, thirdly, it had to supply a set of interesting and progressively more challenging—although not overwhelmingly difficult—projects. Furthermore, at least some of these projects had to provide some introduction to geometry and spatial thinking. Finally, I was also looking for a tool that would allow me to follow my students' progress, check their activity, and review their programs.

As it sometimes happens with units that do not exactly fit in a neat curricular category, I did not have much time to devote to this project and I wanted the students to be able to enjoy it, have fun, experience some measure of success, and complete it with satisfaction in a maximum of three class periods (plus the related homework). After some research I settled on *TouchDevelop*, a free online tool designed at Microsoft Research. *TouchDevelop* is an interactive development environment that allows users to create, test, and run programs and to develop Apps on mobile devices such as smartphones, tablets and laptops. It runs completely in-browser and includes three skill levels (beginner, coder, expert) that provide a smooth transition from drag-and-drop blocks to self-created text. My students found the fully-guided tutorials for beginners easy to follow and the projects offered by the site appealing and exciting. To learn more about *TouchDevelop*, I signed up for a free webinar for educators —organised by SimpleK12 (2000 – 2015), an online teacher learning community—that explained the most commonly used features, including how to set up a class.

Class activity

Firstly, I asked my students to sign up with *TouchDevelop* with their school Google credentials. After creating my 'group', I received a code that I passed on to my students so that they could join my class. Once *TouchDevelop* is launched, it opens on a screen with plenty of resources. I directed the students to go the 'Learn' tab and start with the first beginner project, called *First steps with Turtle*. This project allows the students to create a simple App that makes some fun, colourful geometric drawings —refer to Table 1 for samples of student projects 1 and 2 that have been created using *TouchDevelop*.

Table 1: Samples of student projects using *TouchDevelop*.

Project Number	Project Title	Project URL
1	Unusual turtle	http://tdev.ly/axrzc
2	Super-cool turtle	http://tdev.ly/awbjg

The self-paced tutorials in *TouchDevelop* teach students how the code defines every action the turtle (the cursor) completes, how the computer runs the code line by line, what the effect of each instruction is, and even how the for loop allows to repeat code efficiently. The students also learn how a small change of the turning angle (from 90 to 91 degrees) applied recursively creates a very cool design. Students used the beginners' tutorials—with great success—to generate fractal (Koch's) snowflakes (Projects 3 and 4), fractal trees (Projects 5 and 6) and other types of fractals (Project 7). Figures 1 to 7 show a variety of completed project screen shots, and indicate the variety of ways coding can and has been used with spatial reasoning and geometric designs being a focus within the activity.

Table 2: Samples of student projects using *TouchDevelop* beginner tutorials.

Project Number	Project Title	Project URL
3	Exclusive Snowflakes	http://tdev.ly/wmwla
4	Terrific Snowflakes	http://tdev.ly/sqtcc
5	Tree	http://tdev.ly/grsuc
6	Marvellous Tree	http://tdev.ly/icmt
7	Mind-blowing Fractals	http://tdev.ly/akuba

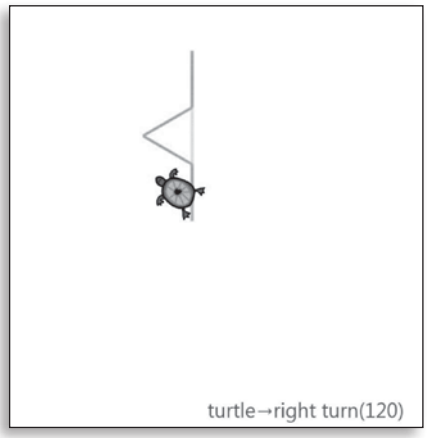

Figure 1: Project 3, an example of coded turning angles and visual output.

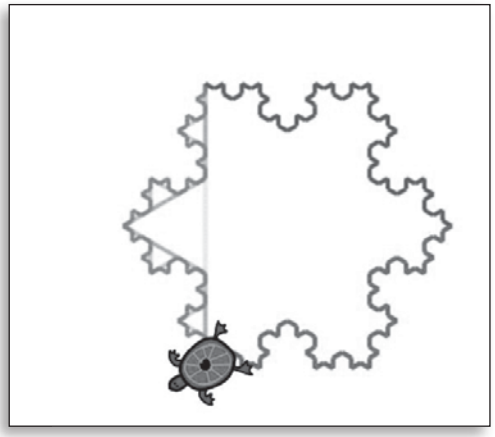

Figure 2: Project 3, 'Exclusive Snowflakes'.

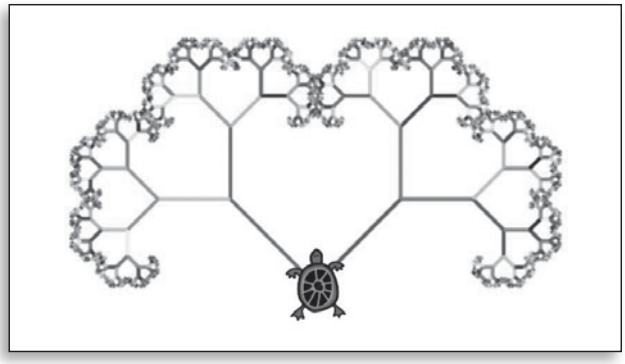

Figure 3: Project 5 'Tree'.

As they progress to exercises involving more written code and reasoning, students will find that they have to work on their geometry skills. For example, they will need to calculate the angles of rotation for the turtle's path, and also figure out how they want to complete some of the paths that the turtle will create. Table 3 summarises some of the sample App projects that students developed, and which link spatial reasoning to geometric design.

Table 3: Samples of student app projects.

Project Number	Project Title	Project URL
8	Mind-boggling App	http://tdev.ly/qrvda
9	Startling App	http://tdev.ly/kquk
10	Magical App	http://tdev.ly/uboxe

Graphing skills (refer to Table 4) were developed further through a series of different exercises completed by students. A project assisting a bear to complete a quest was also a project option selected by some students (refer to Table 5). Figure 6 shows screen shots of the game interfaces coded by students for this quest.

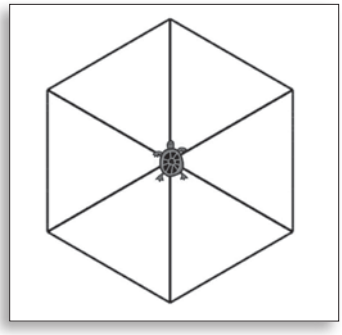

Figure 4: Project 10 'Magical App'.

Table 4: Samples of Student Graphing projects.

Project Number	Project Title	Project URL
11	Weirdness	http://tdev.ly/kwqh
12	Glorious Art	http://tdev.ly/oople
13	Pixel Magic	http://tdev.ly/yyusa
14	Awesome Art	http://tdev.ly/uubh

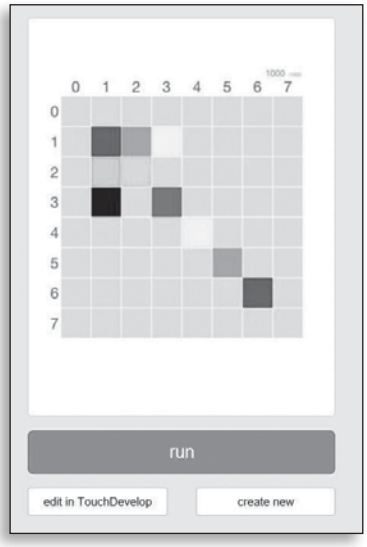

Figure 5: Project 13 'Pixel Magic'. A completed graphing project.

Table 5: Bear Quest projects.

Project Number	Project Title	Project URL
15	Fabulous Quest	http://tdev.ly/autf
16	Bear-ing the Quest	http://tdev.ly/cxofc
17	Unusual Quest	http://tdev.ly/stcca

Figure 6: Sample Bear Quest project interfaces

Simple coded games

Students demonstrated that they can also program a variety of simple games (projects 18 to 20), including a *Pong* type game (Projects 21 to 23). This last game type is more challenging than others created, as it required students to build an object of the right dimensions for the game to be enjoyable. Figure 7 shows an example of the game screen that has been coded.

Some examples of the simple games created by students have been summarised in Table 6. Hint: For the simple game projects in Table 6, you need to click the space bar to make the cat (or monkey) jump, and you need to hit it quickly as soon as the game starts.

Table 6: Example game projects.

Project Number	Project Title	Project URL
18	Astounding Bird	http://tdev.ly/sdkdc
19	Phenomenal Bird	http://tdev.ly/knuda
20	Flappy Bird: Fun Fish	http://tdev.ly/idewa
21	Weird Pong	http://tdev.ly/qwhha
22	BeachDay	http://tdev.ly/aonlc
23	Marvellous Pong	http://tdev.ly/klmca

Figure 7: Example game screen for project 19.

Discussion

Students worked on the programs individually and at their own pace, but they could, and did, discuss and collaborate to overcome the various difficulties and frustrations that arise when writing code. Most girls in my class had no previous experience with programming and at first they were cautious, concerned that coding was going to be too difficult, was not for them and/or was not interesting. Certainly, there were many questions during the first half hour of the initial practical lesson, as the students set up their accounts and started playing with the first coding project. However, this quickly subsided when the students realised that they could answer their own questions by simply trying again, following directions more carefully, problem solving and discussing strategies with their classmates. The hands-on approach and novel, fun projects encouraged a remarkable persistence in solving the numerous problems that the students encountered. This determination in problem solving was possibly one of the most important by-products of this unit and will hopefully transfer to problem solving in other areas of the curriculum.

My students ended up loving this project. They enjoyed working at their own pace and appreciated that it was such a hands-on activity. They liked the collaborative aspect of problem solving together when an obstacle presented itself, while at the same time they could work individually on their projects and produce their own creation. As a tool, *TouchDevelop* is very user-friendly for this age group and it is an easy and engaging way to introduce students to programming concepts. The projects presented were motivating and the students enjoyed being able to create fun Apps and games quickly, regardless of their skill level.

Evaluation and assessment of the activity and student application

I did not write a formal assessment for this lesson, although this could be easily done with a rubric. Teachers could develop an assessment rubric using the electronic tool *RubiStar* (2000-2008) or alternatively create their own. Using a rubric, as a quick assessment tool, would provide constructive feedback to students about their coding project. It would also assist in providing a summary of an individual student's spatial reasoning and application of selected geometry skills. However, as this was the first time I ran this project and the students were completely new to coding, I preferred to look at their experience holistically. Therefore, I graded their portfolio (that I could access online through the group I had created) simply on completion, and then assessed the students' performance (in class work and home work) on their problem solving skills, their ability to collaborate with others and communicate mathematical concepts and ideas, their resilience, their determination, their capacity to overcome frustration and tolerate failure, and their work ethic – the so called 'soft skills' that are just as important for success. To support my observations and record keeping, I kept a tool (a clipboard, a notebook, or an *iPad*) with me at all times to take quick notes about each student's attitude, mindset, and contributions. These progress notes were taken into consideration when writing the report cards, as I included a short written paragraph qualitatively assessing student accomplishments in this activity.

Conclusion

This coding project was valued by the students, who gave it high marks and have been demanding more ever since. It was clear that they had fun and they learned a considerable amount. The students also appreciated the change of pace provided by working independently at their own pace, whilst at the same time having the opportunity to collaborate with their classmates, given that this can be quite rare in a mathematics class. As beginner coding students, the girls particularly enjoyed being able to create Apps and games. They found the various projects accessible and fun, which greatly motivated and engaged them. The students' enthusiastic response and upbeat attitude, together with the positive learning outcome leads me to conclude that this is a valuable activity that provides an intuitive and logical introduction to basic concepts of coding while practicing spatial reasoning and the application of some geometry skills.

References

Australian Curriculum, Assessment and Reporting Authority (ACARA). (2014). *Australian Curriculum: Mathematics.* Retrieved 10 October, 2015 from http://www.australiancurriculum.edu.au/mathematics/curriculum/f-10?layout=1

Australian Curriculum, Assessment and Reporting Authority (ACARA). (2014a). *Australian Curriculum: Technologies.* Retrieved 10 October, 2015 from http://www.australiancurriculum.edu.au/technologies/introduction

Microsoft TouchDevelop. (2015). *TouchDevelop Launch Editor.* Retrieved 12 October 2015 from https://www.touchdevelop.com

National Council of Teachers of Mathematics (NCTM). (2000). *Principles and Standards for School Mathematics.* Reston, VA: National Council of Teachers of Mathematics. Retrieved 12 October 2015 from http://www.nctm.org/standards/

Rubistar. (2000–2008). *Rubistar: Create Rubrics for your Project-Based Learning Activities.* Retrieved 12 October 2015 from http://rubistar.4teachers.org/index.php

SimpleK12. (2000–2015). *SimpleK12: Professional Development in Your Pajamas.* Retrieved 12 October 2015 from http://www.simplek12.com

The College Board. (2015). *AP Data—Archived Data 2014—Research.* Retrieved 12 October 2015 from http://research.collegeboard.org/programs/ap/data/archived/ap-2014

University of Idaho. (2015). *Investigating Influences in Idaho STEM Education: Project Reports & Data.* Retrieved 12 October 2015 from http://www.uidaho.edu/research/STEM/stem-micron/micronstemed/project-reports